Universidad Zaragoza
1542

Trabajo Fin de Grado

Viabilidad de las Contra-Medidas Electromagnéticas (ECM) mediante Radio Definida por Software (SDR)

Autor

Diego García Sánchez

Director/es

Director académico: Dr. D. Francisco Javier Luzón Marco
Director militar: Tte. D. Mikel Ramos Ruiz

Centro Universitario de la Defensa-Academia General Militar

Año 2019

Agradecimientos

Quisiera expresar mi gratitud en primer lugar a todos los componentes del Regimiento de Guerra Electrónica 31, del Batallón de Guerra Electrónica y mas concretamente a la Compañía de Telecomunicaciones, Teniente, suboficiales y personal de tropa, sin cuyo apoyo y ayuda la realización de este trabajo no habría sido posible y con la que espero tener la capacidad de seguir desarrollando estas tecnologías en el futuro.

Me gustaría agradecer también a mis compañeros de practicas en el desarrollo de proyectos en la unidad, en los cuales pude apoyarme siempre que la ocasión lo requería y cuya opinión y consejo han contribuido a que este trabajo llegue hasta donde ha llegado.

A aquellos profesores del CUD que se esfuerzan por enseñar a los cadetes los conocimientos necesarios para no solo la vida militar, sino para nuestro desarrollo como personas. Y mas concretamente a mi director académico que ha conseguido guiarme en el desarrollo de esta memoria.

Finalmente, como no, agradecer a mi familia y amigos el apoyo que me han dado toda la vida y por prestarme un oído cada vez que alguna dificultad de este trabajo se ponía por delante.

A todos vosotros, gracias.

Resumen

Viabilidad de las Contra-Medidas Electromagnéticas (ECM) mediante Radio Definida por Software (SDR)

En la actualidad, el espectro electromagnético, ocupa un gran papel en la resolución de conflictos simétricos y asimétricos, tanto por la importancia táctica que supone a las telecomunicaciones propias, por las que supone para el enemigo. Por esta misma importancia que supone, el espectro electromagnético, se ha convertido un recurso que proteger y aprovechar, y en lo que se centrará el desarrollo de este proyecto, impedir el aprovechamiento del mismo por parte del enemigo.

En este proyecto se van a estudiar y desarrollar las capacidades que impidan el uso del enemigo del espectro electromagnético mediante el uso de radios definidas por software, las cuales son unos dispositivos electrónicos que permiten el procesado de información de señales electromagnéticas y la emisión de las mismas con unas prestaciones de flexibilidad y adaptabilidad que las radios convencionales, las cuales están basadas en hardware, no ofrecían.

Para ello, se procederá al desarrollo de software que permita el uso adecuado de estos dispositivos cumpliendo con los requerimientos y necesidades que el ejército de tierra exige, tanto en eficacia, como en facilidad de uso. Posteriormente se investigará y evaluará la integración de estos dispositivos en los medios de los que actualmente dispone el Ejército de Tierra dedicados a la perturbación de espectro electromagnético, analizando su eficacia conjunta y observando líneas futuras de desarrollo para ambas tecnologías.

Todo ello se realizará mediante el uso de distintas herramientas software de acceso libre y gratuito además de las herramientas de radio definida por software de las que disponen las unidades de guerra electrónica del Ejército de Tierra como el REW 31.

Abstract

Feasibility of Electromagnetic Countermeasures (ECM) through Software Defined Radio (SDR)

Nowadays, the electromagnetic spectrum plays a major role in resolving symmetrical and asymmetrical conflicts, both because of the tactical importance it poses to telecommunications itself in our own use, as well as, for those it poses to the enemy. Because of this importance, the electromagnetic spectrum has become a resource to protect and exploit, and in which the development of this project will focus on, to prevent the enemy from exploiting it.

This project will study and develop the capabilities of preventing the use of the enemy of the electromagnetic spectrum through the use of software-defined radios, which are electronic devices that allow the processing of information from electromagnetic signals and their emission with the performance of flexibility and adaptability that conventional radios, which are software based, did not offer.

To do this, we will develop software that allows the proper use of these devices in compliance with the requirements and needs that the Spanish Army requires, both in efficiency and in ease of use. The integration of these devices into the media currently available by the Spanish Army that is dedicated to electromagnetic spectrum disturbance will be investigated and evaluated, analyzing their joint effectiveness and observing future lines of development for both technologies.

All this will be done through the use of various free and free access software tools in addition to the software-defined radio tools available to the Spanish Army's electronic warfare units such as the REW31.

Índice de figuras

Índice de siglas y acrónimos

Siglas / Acrónimos	Significado en Inglés	Significado en Español
SDR	Software-Defined Radio	Radio Definida por Software
EW	Electronic Warfare	Guerra Electrónica
ESM	Electronic Support Measures	Medidas de apoyo Electrónico
ECM	Electronic Countermeasures	Contra-Medidas Electrónicas
EPM	Electronic Protection Measures	Medidas de Protección Electrónica
TX	Transmission	Transmisión
RX	Reception	Recepción
GPS	Global Positioning System	Sistema de Posicionamiento Global
GNSS	Global Navigation Satellite System	Sistema Global de Navegación por Satélite
NAVSTAR	NAVigation Satellite Timing and Ranging	//
A-GPS	Assisted GPS	GPS Asistido
ET	//	Ejército de Tierra
REW 31	//	Regimiento de Guerra Electrónica Nº31
DSP	Digital Signal Processor	Procesador de Señales Digitales
DAC	Digital Analog Conversor	Conversor Digital Analógico

Índice

1. Introducción

1.1. Motivación

El primer caso documentado de medidas de Guerra Electrónica data de la guerra ruso-japonesa de 1904-1905, cuando un almirante de la flota de navíos rusa propuso impedir las comunicaciones radio del enemigo a base de transmitir una señal con mayor potencia para interferir en la recepción enemiga. Desde aquel primer caso documentado a principios del siglo XX el concepto de guerra electrónica siguió en los ejércitos, y fue durante la Segunda Guerra Mundial y la Guerra Fría cuando se reconoció el potencial que esta aportaba.

La guerra electrónica fue desarrollándose a medida que lo hacían las medios de comunicación electromagnéticos, evolucionando y especializándose ambas simultáneamente, hasta el punto de que era necesario un dispositivo diferente para cada acción de guerra electrónica. Este desarrollo dio lugar a la necesidad de un amplio abanico de dispositivos hardware específicos que suponían y actualmente suponen un gran coste al ejército tanto en el propio material como en la instrucción de los militares que lo operan.

La solución a todos estos problemas la propuso la llegada de las radios definidas por software (SDR," Software Defined Radio"), las cuales han conseguido abrir un nuevo enfoque a la guerra electrónica. Estos dispositivos que contienen un hardware muy genérico y de bajo coste, son capaces de obtener una gran flexibilidad gracias a que permiten ser configurados a través de un ordenador en función de las necesidades del usuario.

1.2. Objetivos del proyecto

En relación con la idoneidad del uso de las SDR en la guerra electrónica, el objetivo principal de este proyecto es explorar la posibilidad del empleo de las SDR para llevar a cabo contramedidas electrónicas que impidan o dificulten el uso del espectro electromagnético por parte del enemigo. Además, se estudiará la integración de los sistemas SDR en los sistemas de contramedidas electrónicas con los que cuenta actualmente el ejército.

Para llevar a cabo estos objetivos principales se plantean tres objetivos secundarios:

- Perturbación: En este bloque se desarrollará un software que permita la generación de señales mediante SDR en gran parte del espectro electromagnético de telecomunicaciones (1MHz a 6 GHz) con el fin de perturbar la recepción de un dispositivo.
- Decepción o engaño: Conseguir el "engaño" mediante una señal electromagnética es bastante más complejo que su simple perturbación, ya que se debe tener en cuenta las particularidades técnicas de la información que transmite dicha señal, no pudiéndose desarrollar un software genérico para la decepción de cualquier tipo de señal electromagnética. Por ello, este objetivo secundario se centra en un solo tipo de señal, la señal de GPS. Para ello se realizará la simulación de los datos enviados por los

sistemas satélite, para, con el desarrollo de software adecuado, su posterior emisión a través del SDR con la finalidad de engañar los dispositivos GPS. Este tipo de decepción es más comúnmente conocido como GPS Spoofing.

- Integración: se analizarán las modificaciones Hardware y Software necesarias para la utilización de los dispositivos SDR en las Estaciones Perturbadoras (EP) del Ejército de Tierra y las posibles ventajas e inconvenientes que estos podrían suponer en las mismas con relación a la eficacia, fiabilidad y requerimientos futuros del sistema final.

1.3. Estructura de la memoria

La memoria se organiza en siete bloques, comenzando por el bloque **Introducción** en el cual se explican las motivaciones de las que surge la investigación de las capacidades de los dispositivos SDR y la necesidad del desarrollo de programas para los mismos. Posteriormente, se enumeran los objetivos necesarios para cumplir los propósitos de este proyecto y finalmente indicar como se va a estructurar la memoria.

En el apartado **Estado del Arte**, se explican de forma breve los conocimientos básicos necesarios para comprender los fundamentos sobre los que este proyecto se asienta. En él se expondrá el concepto de Guerra Electrónica y las diferentes finalidades que esta puede tener en función de las necesidades del ejército; se analizarán los sistemas SDR explicando sus características básicas y las diferencias que poseen unos sistemas de otros; se explicará el dispositivo elegido para el desarrollo de este proyecto, el HackRF One, junto a sus especificaciones por las que ha sido seleccionado y por último, se explicarán las bases de funcionamiento de los sistemas GPS y como estos calculan la posición de un dispositivo tanto de forma tradicional como mediante el uso de tecnologías más actuales.

El tercer apartado, **Metodología**, relata de forma breve las actividades realizadas, las necesidades exigidas y las herramientas utilizadas (como el software GNU Radio para el desarrollo de programas), con las que se ha llevado a cabo el desarrollo del proyecto

Posteriormente en el bloque **Perturbación del espectro electromagnético** que es el referente al primer objetivo del proyecto. En este apartado se desarrollará el concepto de acción de perturbación o *Jamming* y las necesidades del programa que ejecutará estas acciones tanto en lo referente a explotación de las capacidades de los dispositivos SDR, como de facilidad de manejo por el usuario, y posteriormente, el proceso de desarrollo del programa y detalles de los resultados prácticos obtenidos.

El quinto bloque, **Decepción de dispositivos GPS (GPS Spoofing)**, comprende la parte correspondiente al segundo objetivo del proyecto, en la cual vendrán detallado el modo de generación de los archivos de simulación GPS, explicando las capacidades de realizarse tanto de forma estática como dinámica. Posteriormente se muestra el diseño realizado para la creación del programa emisor de la simulación GPS y los resultados obtenidos en los distintos tipos de receptores utilizados además de las causas de los fallos en algunos de los experimentos prácticos.

El siguiente aparatado, **Integración de medios**, relata las modificaciones necesarias en el sistema EP del ejército para la correcta integración del sistema SDR, además de las medidas a tener en cuenta para la protección de los amplificadores de señal que utiliza. Posteriormente, analiza los resultados obtenidos de la perturbación resultante a la integración de los dos medios, y hace un análisis de los posibles problemas a resolver para la integración de forma oficial y a nivel de los estándares de calidad y capacidades del ET

Finalmente, en el apartado **Conclusiones** se evaluará la viabilidad del uso de los SDR en las unidades de EW teniendo en cuenta el conjunto de posibilidades y capacidades que ofrecen frente a los sistemas actuales del ET, además de la previsión de las modificaciones mínimas necesarias para su adecuada implantación oficial en el ejército.

2. Estado del Arte

2.1. Guerra Electrónica

El término Guerra Electrónica (EW) es el nombre asociado al conjunto de actividades tanto tecnológicas como electromagnéticas, que tienen la finalidad de reducir o impedir la utilización del espectro electromagnético por parte del enemigo al tiempo que conserva la disponibilidad del mismo en beneficio propio[1].

Actualmente los equipos electrónicos son imprescindibles en todos los niveles del combate y son una herramienta fundamental tanto para la coordinación y el control de las unidades como para los elementos de navegación y adquisición de blancos. El hecho de que todos estos sistemas y herramientas dependan en gran medida del espectro electromagnético convierte a la EW en un elemento indispensable en lo relativo a inteligencia y operatividad.

El término EW engloba a muchas acciones diferentes que, debido a la complejidad de las operaciones militares, se categorizan en tres tipos fundamentales:

- Medidas de apoyo a la Guerra Electrónica (ESM): Este área comprende a las acciones de EW cuya finalidad es la obtención de información acerca del espectro electromagnético con la intención de conocer datos sobre una fuente de energía (localización de origen, información trasmitida, monitorización, etc.)

- Medidas de Protección Electrónica (EPM): Engloba a todas las medidas realizadas con la intención de proteger nuestro uso del espectro electromagnético a pesar de las medidas de EW que el enemigo pueda utilizar.

- Contramedidas Electrónicas (ECM): ECM se refiere a las acciones destinadas a impedir o reducir la correcta utilización del espectro electromagnético por parte del enemigo.

Como se ha comentado en la introducción, este proyecto se centra en este último grupo, más concretamente en las medidas de perturbación electromagnética, y la decepción del enemigo mediante el falseamiento de señales.

Actualmente el Ejército de Tierra cuenta con un sistema montado sobre vehículo con la capacidad de realizar acciones EW, más concretamente las del tipo ECM. Este sistema son los denominados EP y son sobre los cuales se basará la integración de los dispositivos de radio definida por software. El grado de confidencialidad de estos sistemas, impide la difusión de algunos de sus detalles técnicos, y capacidades en las acciones ECM con lo que la información aportada a este proyecto sobre ellos será limitada.

2.2. Radio Definida por Software (SDR)

Software-Defined Radio (SDR) es el término utilizado para referirse a los dispositivos de comunicación por radiofrecuencia en los que las funciones de procesamiento de señales son realizadas por un Software en lugar de por elementos Hardware, como ocurre en los dispositivos radio tradicionales[2].

Los dispositivos SDR ofrecen un nivel de flexibilidad mucho mayor a su contraparte más tradicional. El hecho de que gran parte de los componentes de un SDR sean manejados por software implica que gran parte de las funcionalidades que el sistema ofrece puedan ser refinadas, moduladas o modificadas simplemente descargando un nuevo programa o hasta modificando el software ya existente.

Los SDR tienen una serie de características y especificaciones que definen sus márgenes de uso y según estas capacidades, las funciones que desempeñan varían enormemente. Entre estos dispositivos se encuentran modelos que están especialmente diseñados para el análisis del espectro electromagnético, para la interceptación de señales radio o la emisión de las mismas o incluso como componentes en proyectos de domótica.

Las principales especificaciones limitantes a la hora de conocer las funciones que podrá desempeñar un SDR son las siguientes:

-Rango de frecuencias: Es la parte del espectro electromagnético en el que el dispositivo es capaz de procesar[1] los datos que recibe. Se mide en Hz.

-Tx/Rx (Transmisión/Recepción): Indica las capacidades del SDR de recibir señales de radio, emitirlas o realizar ambas con un mismo dispositivo.

-Ancho de banda: Es la banda de frecuencias que un SDR es capaz de analizar de forma simultánea. Se mide en Hz.

-Error de reloj[2]: Indica el desfase del reloj interno del SDR que afectará a la precisión con la que trabaja en diferentes frecuencias. Se mide en ppm. (Partes por millón)

-Potencia máxima de emisión: Indica la intensidad/potencia con la que un dispositivo es capaz de emitir una señal; este valor puede variar en función de la frecuencia. Se mide en W o en dB.

2.3. Dispositivo empleado: HackRF One

El HackRF One es un dispositivo SDR nacido en 2014 a través de una campaña KickStarter[3] y que destaca por el amplio abanico de capacidades que ofrece. Por un precio de menos de 120€ el HackRf One es un dispositivo con un rango de frecuencias de trabajo que van desde los 1MHz a los 6GHz y un ancho de banda de recepción de 10MHz de trabajo, un reloj interno con un error de 20ppm, además de la capacidad de tanto recibir como emitir señales de radiofrecuencia con una potencia máxima de emisión teórica de 30mW a 1mW dependiendo de la banda de frecuencia[3].

---

[1] Aunque un SDR tenga la capacidad de procesar datos en una frecuencia es necesaria una antena capaz de recibir/emitir en la misma banda de trabajo para su correcto funcionamiento.

[2] Todos los SDR tienen un reloj interno que les da una señal periódica con la que procesan las señales de radio recibidas.

[3] KickStarter es una conocida página web de crowdfunding, la cual ha conseguido financiar una amplia gama de proyectos creativos alrededor del mundo.

Estas capacidades y su precio relativamente bajo lo han hecho un dispositivo muy popular entre hackers, radioaficionados e investigadores de fuerzas de seguridad, lo que ha causado que tenga una comunidad dedicada al desarrollo de software dedicado para este dispositivo.

Debido a que las capacidades técnicas se ajustan en gran medida a las requeridas y la comunidad de usuarios de este dispositivo aportan una gran cantidad de conocimiento de fácil acceso el HackRF One será el dispositivo sobre el cual se centran las acciones ECM del proyecto.

Figura 1

Imagen del dispositivo HackRF One. Fuente: AMAZON

2.4. Sistemas Global de Navegación por Satélite (GNSS)

El concepto de GNSS (Global Navigation Satellite System) se refiere a la constelación de satélites que orbitan la Tierra y que tienen la capacidad de emitir a una frecuencia determinada con la finalidad de que un receptor en la superficie sea capaz de recibirla y calcular con ella su posicionamiento.

Nacido como un prototipo experimental, en la década de los 80 se desarrollaron y lanzaron los primeros satélites que dieron el primer servicio de posicionamiento global. NAVSTAR-GPS (más conocido como GPS) fue el nombre dado a este primer sistema de posicionamiento global con el cual, se esperaba establecer un método global para la localización de objetos en las tres dimensiones espaciales más la localización temporal[4].

Aunque originalmente la creación de este sistema de navegación era con intención de su uso en el ámbito militar, rápidamente este sistema se puso a disposición del público civil y es hoy en día una herramienta utilizada globalmente. Es esta universalización del sistema GPS, la causa por la que nos centraremos en él a lo largo del proyecto.

El proceso por el cual los dispositivos GPS son capaces de calcular la posición es relativamente sencillo y consta de dos pasos fundamentales:

Inicalmente, los satélites GPS emiten en una frecuencia determinada (1.57542 Ghz en la mayoria de los casos) un código en formato binario llamado efemérides. Este código se compone de un indicador que identifica el satélite que esta envíando la señal; la posicion exacta del satélite en el momento de la emisión de la señal y la hora exacta en horas, minutos, segundos y milisegundos en el que envía esta señal.[5]

Una vez recibidas las efemérides, el receptor desencripta la señal y la compara con el tiempo de la recepción, de esta forma el receptor es capaz de saber el tiempo que ha estado viajeando la señal y sabiendo que esta ha viajado a una velocidad constante[4] es capaz de calcular la distancia esferica a la que se encuentra del satélite[6].

- Utilizando este método, con un satélite, el receptor es capaz de calcular la esfera de posición en la que se encuentra.

- Con dos satélites, el dispositivo es capaz de reducir su posición a la circunferencia que surge de las dos esferas.

- La recepción de un tercer satélite da los dos puntos posibles en los que se encuentra nuestro receptor.

- Finalmente es con el cuarto satélite con el que el dispositivo es capaz de discernir cuál de los dos puntos es en el que se encuentra.

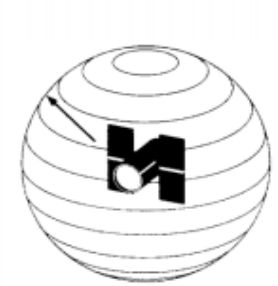

Figura 2

Esfera de posicionamiento con un satélite. Fuente: CUD-AGM SIGTEL

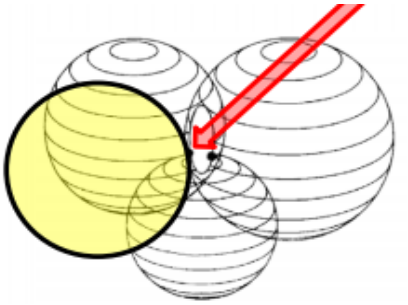

Figura 3

Posicionamiento establecido con cuatro satélites. Fuente: CUD-AGM SIGTEL

Actualmente existen tres constelaciones de satélites que proporcionan posicionamiento con cobertura global: el “GALILEO positioning system” de la Unión Europea, el sistema GLONASS operado por el Ministerio de Defensa de la Federación Rusa y el sistema NAVSTAR-GPS operado por el Ministerio de Defensa de los EE. UU.

[4] La velocidad de transmisión de la señal no es exactamente la misma debido a diferencias en las distintas capas de la atmosfera, pero los errores de precisión que estas pueden causar no superan los 10m.

De estos tres sistemas el más usado tanto a nivel global como en el territorio español es la constelación NAVSTAR-GPS y es en la que se centrarán las medidas GPS Spoofing de este proyecto. Este sistema utiliza dos bandas de trabajo (L1 a 1.57542Ghz y L2 a 1.22760Ghz), con cuatro y tres códigos de cifrado respectivamente, en función de si los servicios que ofrecen son civiles o militares[7].

Aunque la banda de trabajo civil L1 del NASTAR-GPS es capaz de ofrecer precisiones de hasta 2 metros en condiciones óptimas, y la de encriptación militar ofrece hasta menos de un metro de error; la mayoría de los dispositivos inteligentes (smartphones/tablets) en los que se centra este proyecto utilizan la tecnología A-GPS (Assisted GSP) para conseguir un aumento en la precisión, así como en la rapidez en el posicionamiento.

La tecnología A-GPS utiliza las antenas de telefonía móvil a las que se conectan los dispositivos móviles para dar una ubicación aproximada en un radio de pocos kilómetros; además de esto indica al receptor GPS los satélites con los que debería hacer contacto a esa hora del día y en esa zona determinada para evitar tiempos de búsqueda innecesarios.

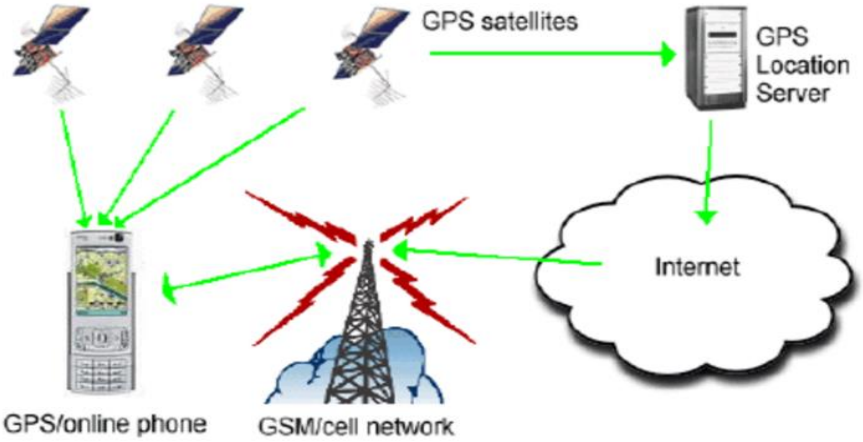

Figura 4

Esquema de flujo de información en la tecnología A-GPS
https://www.researchgate.net/figure/Assisted-Global-Positioning-System-A-GPS_fig5_287156709

3. Metodologia

Como ya se ha explicado previamente, el proyecto gira en torno al dispositivo HackRF One y las capacidades de emisión que tiene en comparación con los sistemas actuales del ET. Para ello se requiere que el dispositivo sea capaz de igualar las capacidades de las EP y en la medida de lo posible superarlas. Observando simplemente las especificaciones de ambos dispositivos, se puede afirmar que la radio HackRF One tiene una banda de frecuencias de trabajo que engloba de forma completa a aquella de la EP. De la misma forma, el HackRF One también es capaz de emitir en anchos de banda de hasta 56MHz que teóricamente es superior a aquel de la EP.

Tanto la perturbación de señales, como la decepción GPS, se han realizado a través del uso de aplicaciones Software de base Linux y del desarrollo de programas en código Python, los cuales debían tener la capacidad de cumplir con las necesidades específicas de las unidades de Guerra Electrónica. Estas necesidades incluyen tanto la flexibilidad de trabajo, como la facilidad de uso para usuarios sin conocimientos previos en el uso del sistema operativo Linux.

Para este desarrollo se utilizó el programa de SDR llamado “GNU Radio” el cual es una herramienta de desarrollo de software de código abierto, en nuestro caso en base Linux, que permite la generación de diagramas de procesado en forma de bloques tanto para uso real con SDR como para simulaciones sin Hardware[8][9]. Este programa se basa en la creación de diagramas de flujo mediante bloques con capacidades lógicas o matemáticas, que cuando conectados entre sí permiten el procesado de la información recibida a través de un dispositivos SDR, o como es el caso de este proyecto, la emisión de los datos procesados por un ordenador[10].

Por último, el estudio de la integración se analizarán las modificaciones Hardware y Software necesarias para la utilización de los dispositivos SDR en las Estaciones Perturbadoras y se realizarán las comprobaciones de funcionalidad mediante la observación del espectro electromagnético de la misma forma que se utilizó para evaluar las capacidades de perturbación del dispositivo HackRF One.

4. Perturbación del espectro electromagnético.

Como se ha explicado previamente, el primer objetivo de este proyecto es el desarrollo de software para SDR que permita la perturbación o "Jamming" del espectro electromagnético. El jamming es una forma de impedir la comunicación entre dos dispositivos de radiofrecuencia en la cual un tercero emite una señal que le llegue con más potencia al receptor en las frecuencias o en alguno de los canales de frecuencia en los que se establece la comunicación, de esta forma la señal perturbadora, "ensordece" al receptor haciendo ineficaz la transmisión de información entre el emisor y el receptor[11].

En el "jamming" el éxito de las perturbaciones en radiofrecuencia radica en la potencia de la emisión con la que se envía la señal de perturbación, sin importar la información que se está enviando, ya que basta con transmitir una señal de ruido al azar. Sin embargo, a la hora de perturbar una o varias señales de radio sí que hay que tener en cuenta las características técnicas de la señal que se va a emitir, ya que detalles como la frecuencia central de la emisión y el ancho de banda de la misma afectarán a la potencia que el dispositivo es realmente capaz de distribuir. Los dispositivos SDR capaces de emitir como el "HackRF One" no tienen la misma potencia de emisión en todo el espectro en el que trabaja: por regla general la potencia máxima real de emisión disminuye desde los 30mW en las frecuencias más bajas a 0.1mW en las frecuencias más altas. Asimismo, hay que tener en cuenta que, a mayor ancho de banda de perturbación, es mayor la parte del espectro en la que se distribuye la potencia de emisión, y por lo tanto menor la intensidad del jamming en las frecuencias individuales.

Teniendo en cuenta lo anterior, en este apartado se propone el desarrollo, mediante GNU Radio, de un programa capaz de emitir un archivo de audio (.WAV en este caso) y que facilite al usuario la modificación de:

a) La frecuencia de emisión.

b) El ancho de banda de la emisión.

c) La potencia de perturbación en caso de que fuera necesario.

Además, con la finalidad de facilitar el uso de este programa, también se incorporará una interfaz gráfica sencilla que haga accesible el correcto funcionamiento de este programa a usuarios sin conocimientos previos de lenguaje de programación.

Para poder desarrollar el anterior programa es necesario analizar cómo se produce la transmisión de la información en los dispositivos SDR, para de esta forma saber cómo se tendrá que procesar el archivo de audio antes de enviarlo:

Como se muestra en la figura 5, en un SDR como el HackRF One el procesado de la información comienza cuando el procesador digital de señales (DSP por sus siglas en inglés) recibe el input de datos en código binario que se desea transmitir. Este DSP se encarga de procesar el archivo en tiempo real, generando la señal en forma de números complejos que representa la onda sinusoidal que se quiere emitir. Esta señal generada ya cuenta con las características de la emisión que se quiera transmitir tanto de frecuencia, como de ancho de banda. Tras esto la información pasa a un conversor digital-analógico (DAC según sus siglas en ingles) el cual transforma las instrucciones recibidas por el procesador digital de señales en los pulsos de energía que son enviados a través de la antena[2].

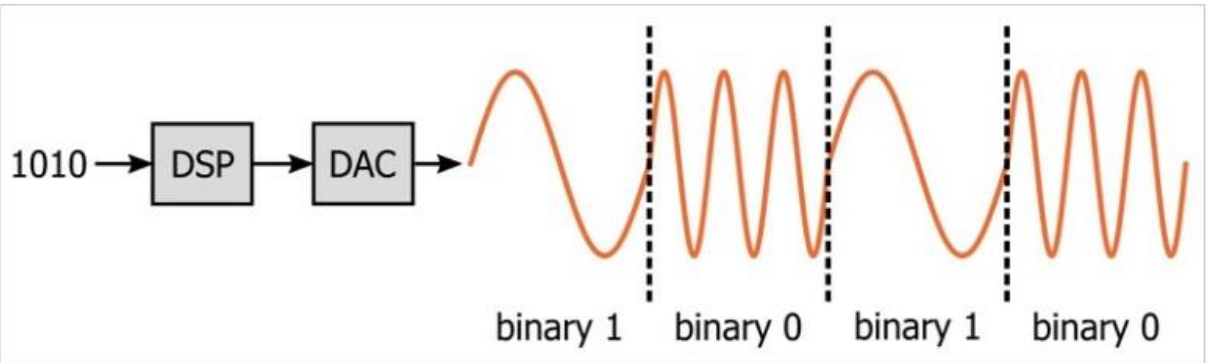

Figura 5

Esquema de la generación de ondas en un dispositivo SDR. Fuente: https://www.allaboutcircuits.com/technical-articles/introduction-to-software-defined-radio/

4.1. Jamming

Como se acaba de ver en el apartado anterior, el procesado de la información se realiza a través del DSP y es el programa generado a través de GNU Radio el que le proporciona las instrucciones necesarias. [12]

El primer programa desarrollado en código Python (Anexo A) viene generado a traves del siguiente diagrama que está compuesto por las siguientes bloques de procesado:

El primer bloque, denominado **File Source**, asigna la ubicación del archivo fuente (en nuestro caso .WAV) que se va a transmitir, el cual es tomado como una vector/lista de números reales. Este vector es posteriormente procesado a través del bloque **Multiply Const,** el cual es usado multiplicar cada uno de los componentes del vector por una constante consiguiendo de esta forma aumentar o disminuir el volumen del audio recibido. En nuestro caso, el multiplicado del vector se realiza con una constante de valor cercano a cero (0.00003) para homogeneizar la emisión a otra que sea lo más similar posible al ruido electromagnético.

Una vez procesado el vector, este sigue estando compuesto por números reales los cuales han de ser transformados en números complejos, ya que estos son la representación numérica de la onda sinusoidal de las ondas de radiofrecuencia. Esto se consigue a través del bloque **WBFM Transmit** el cual recibe un input con un ratio de muestreo (Audio rate) de 44.1KHz que es el estándar de los archivos.WAV, y lo transforma a un output (Quadrature rate) de 88.2KHz para los valores complejos.[5]

Tras esto, el **Rational Resampler** agrupa los bloques de 88.2KHz en tramos de 2.6MHz de ratio de muestreo el cual es el idóneo para la transmisión desde el dispositivo HackRF One y finalmente, el bloque **osmocon Sink** indica el dispositivo USB en el que se van a volcar los datos, y los valores de

[5] En este caso los valores como Tau y Max deviation son para mejorar la calidad de la transmisión que no son relevantes para la perturbación.

potencia (IF Gain/RF Gain), frecuencia de emisión (Frequency) y ratio de muestreo (Sample Rate) con los que se va a transmitir .

Por último, los bloques denominados **WX GUI Slider** son aquellos que proporcionarán la interfaz gráfica y las capacidad de modificar las variables a tiempo real, los cuales permiten trabajar en unos rango de valores predefinidos.

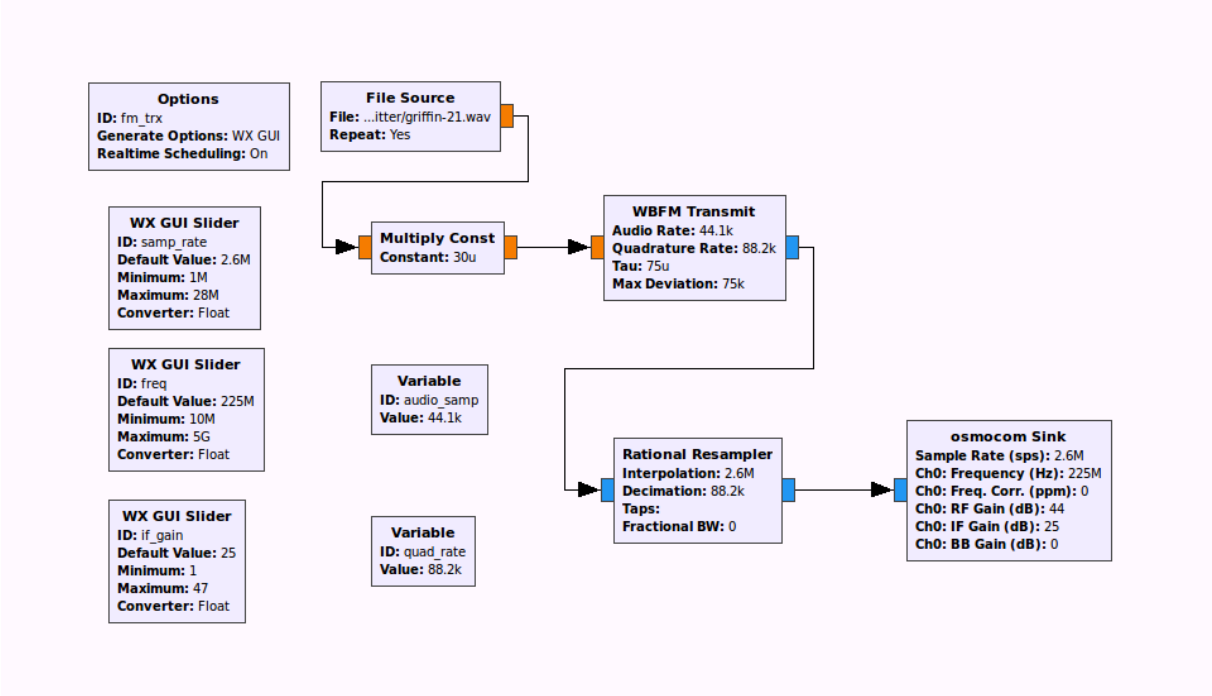

Figura 6

Diagrama de bloques GNU Radio. Transmisión FM.

El diagrama aquí mostrado es el que genera el código en lenguaje Python el cual, al ser ejecutado, inicia la emisión en bucle del archivo .WAV seleccionado, además de generar tres ventanas a través de las cuales se puede modificar a tiempo real: la frecuencia de emisión (entre 10MHz y 5GHz), el ancho de banda (entre 1MHz y 28MHz) y la potencia de emisión (valores entre 1 y 47 establecidos por el HackRF One).[13]

En cuanto a los resultados prácticos obtenidos, las capacidades de perturbación del programa resultaron satisfactorias, consiguiendo una emisión positiva en todas las frecuencias del espectro desde los 10MHz a los 5GHz. Las capacidades de perturbación en frecuencias individuales superan un Signal to Noise Ratio (SNR: diferencia en Db entre el ruido de fondo y el pico de una señal) de unos 50 Db.

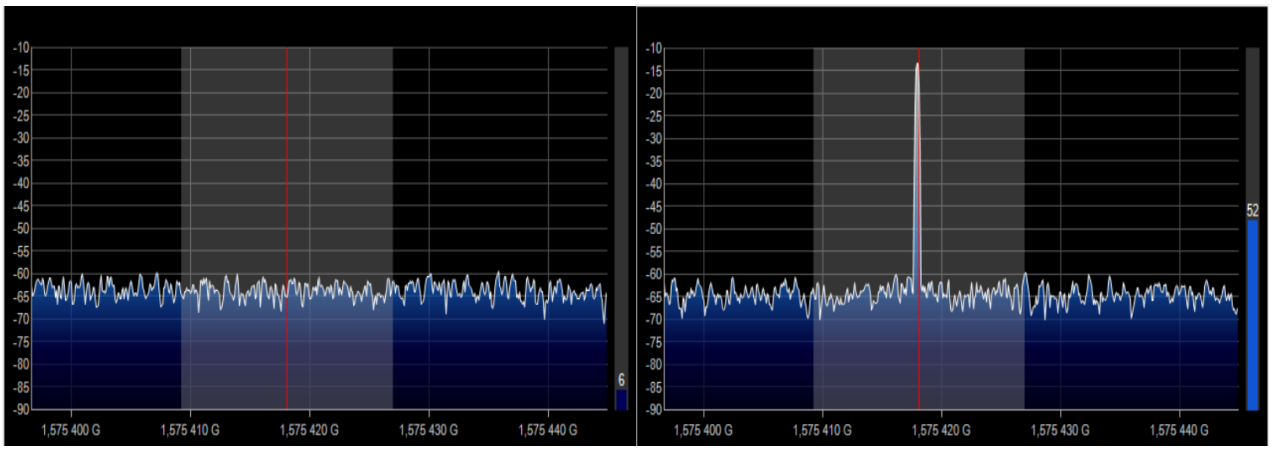

Figura 7

Comparación gráfica del antes y el durante de la perturbación en la frecuencia 1.575418GHz. SNR ~ 50

Además, dentro de la capacidad de variación del ancho de banda de emisión, se obtiene un máximo de potencia en los 1.5MHz de emisión, y con emisiones inestables a partir de los 16MHz en los que la señal comienza a oscilar. (Para observar los resultados de las demas pruebas ANEXO D)

5. Decepcion de dispositivos GPS (GPS Spoofing)

Una vez desarrolladas las capacidades de emisión a través del HackRF One es posible proceder al siguiente objetivo del proyecto, el cual se trata de la decepción de tecnologías GPS mediante SDR. Para ello, la realización del GPS Spoofing se va a realizar en dos pasos:

a) Se va a realizar la generación de información de los satélites que los receptores utilizan para calcular su posición a cada instante de tiempo. El desarrollo de estos datos se realizará a través del uso de un programa ya existente.

b) La emisión de la información será realizada mediante el desarrollo de un programa mediante el software GNU Radio. Este programa, similarmente al requerido para la perturbación del espectro electromagnético tendrá que ser capaz de emitir el archivo generado en el paso anterior y emitirlo en la frecuencia de trabajo de los satélites GPS.

5.1. Generacion de coordenadas

Para la correcta ejecución del GPS Spoofing, en primer lugar, se necesita un programa con la capacidad de simular de forma precisa los datos ficticios de los códigos de posicionamiento y tiempo que los satélites envían continuamente para que el receptor que los recoja sea capaz de calcular su posición, que no será su posición real sino una posición ficticia. Para ello el programa tiene que calcular, a partir de la posición ficticia del objetivo que proporcione el usuario, tanto el momento en el que se producirá la recepción por parte del objetivo, como la trayectoria prevista de los satélites durante un periodo de tiempo, para así obtener la ubicación ficticia en espacio y tiempo de cada uno de los satélites necesarios para engañar al receptor (mínimo 4 satélites, más de 7 para una precisión inferior a +-10m)[4].

A través de la plataforma de intercambio y revisión de código GitHub, se puede acceder a un código de simulación satélite denominado “gps-sdr-sim”[3], basado en el lenguaje de programación C, que es capaz de generar un *stream* de datos FIFO (first in first out)[14]. Este programa solo necesita de forma externa la previsión de los datos de trayectoria de la constelación de satélites para utilizarla como base de datos a la hora de seleccionar que satélites simular y sus posiciones en el momento de uso del programa. Toda esta información es accesible a través de internet mediante un servidor de transferencia de archivos de la NASA, en la cual se publican diariamente varios documentos con la información de los distintos satélites. ftp://cddis.gsfc.nasa.gov/gnss/data/daily/2019/brdc/

Una vez descargado el archivo más reciente, se indica al programa su ubicación dentro del sistema y se ejecuta seleccionando las coordenadas en latitud, longitud y altitud, y el momento de inicio de la simulación; a través de lo cual se genera el archivo binario que contiene los datos simulados del posicionamiento estático del objetivo.

Otra opción del programa es la generación de coordenadas dinámicas, en las que, en lugar de simular una posición fija, el usuario es capaz de predefinir un “camino”. Este “camino”, es posteriormente traducido a varias posiciones estáticas que se van desarrollando a través del tiempo, para simular de esta forma el movimiento del objetivo. Este desplazamiento es definido como una ruta sobre Google Earth, la cual es posteriormente exportada al programa SatGenNMEA[15]. Este programa permite regular valores como velocidad de desplazamiento o aceleración máxima del receptor y exporta un

archivo .CSV. Este archivo está compuesto por una matriz de latitud, longitud, altitud, y tiempo pasado desde el inicio de la simulación, dando así la información del posicionamiento en cualquier instante de la simulación. (ejemplo parcial de archivo .CSV generado en ANEXO C).

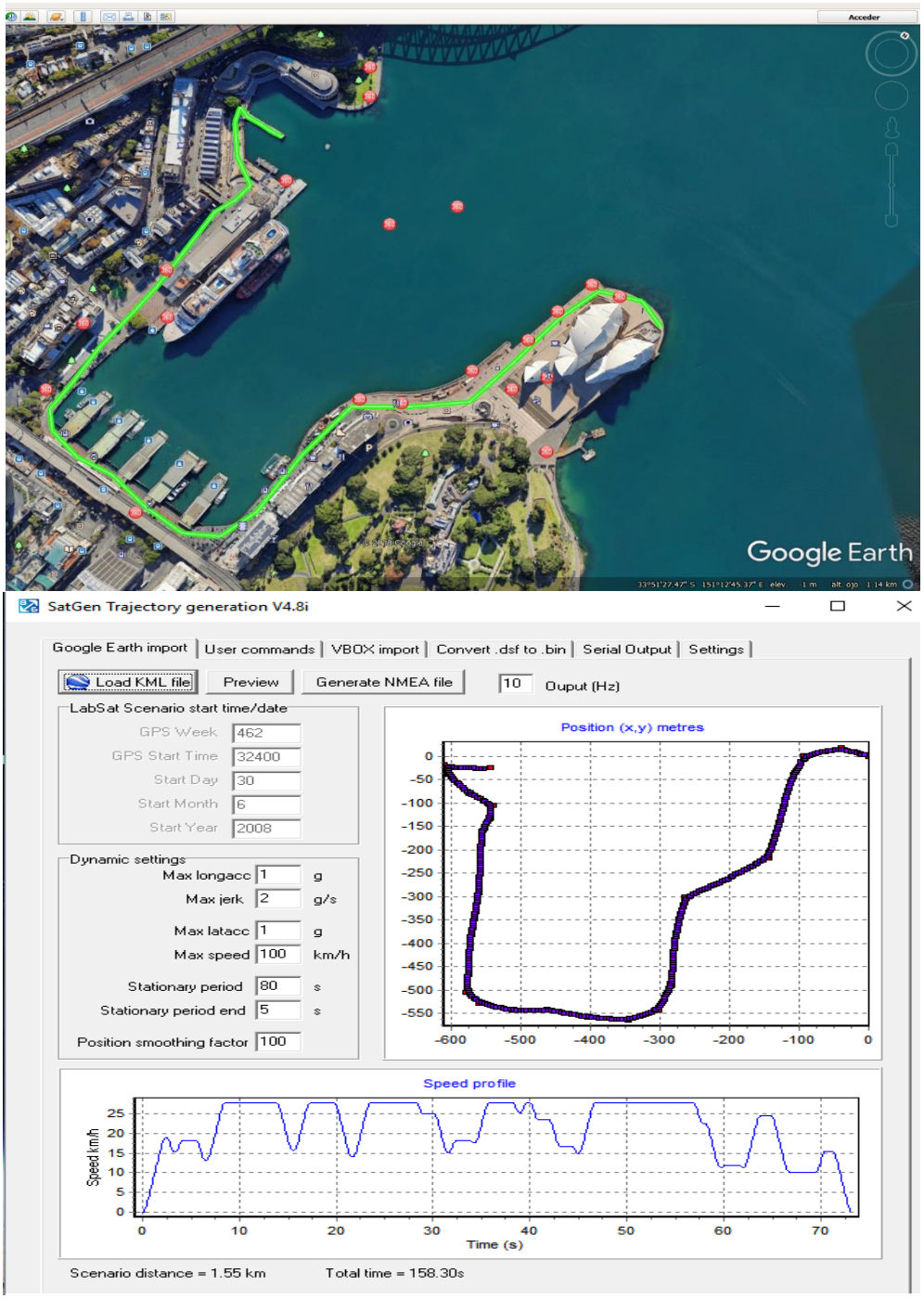

Figura 8

.Ejemplo de generación de recorrido en GPS Spoofing Dinámico.

Debido a que para la realización del GPS Spoofing dinámico requiere la generación previa del archivo binario el sistema impone una limitación a la duración máxima de la simulación en 300s por tamaño del archivo generado (aproximadamente 5GB por un archivo de 300s), a diferencia del GPS Spoofing estático, en

el que le archivo se emite en *streaming* con lo que no hace falta generación previa y por consiguiente ocupa un espacio despreciable.

5.2. Emisión de archivo y pruebas

De forma similar al programa realizado en el objetivo anterior, el programa de emisión de la simulación GPS (ANEXO B) viene generado por el un diagrama de bloques de procesado.

Debido al cambio de formato en el archivo generado anteriormente es necesario realizar modificaciones en el diagrama GNU Radio para la correcta emisión del archivo. Este nuevo formato del archivo a emitir es en código binario que se emite en grupos de 8, es decir en bytes.

En este caso, para su correcta emisión, el flujo de datos se pasa a través del bloque **iChar To Complex**, el cual recibe los bytes de información y devuelve como output a su equivalente en número complejo. Debido a que los datos enviados son información binaria y no audios, no es relevante el ratio de muestreo al que estén agrupada la información, solamente el ratio de muestreo de emisión. Finalmente, estos valores complejos son volcados junto a la frecuencia de emisión y el ratio de muestreo sobre el dispositivo designado por el bloque **osmocon Sink**.

Adicionalmente, al igual que el diagrama de perturbación, se ha incluido los bloques de interfaz gráfica **WX GUI Slider** para el ajuste en directo de valores como la potencia de emisión, y si fuera necesario por el uso de un dispositivo diferente al HackRF One también que permitan el ratio de muestreo de la emisión. También se incluye el bloque **WX GUI FFT Sink** el cual da una imagen de la señal emitida en el espectro electromagnético para establecer comprobación visual de la señal.

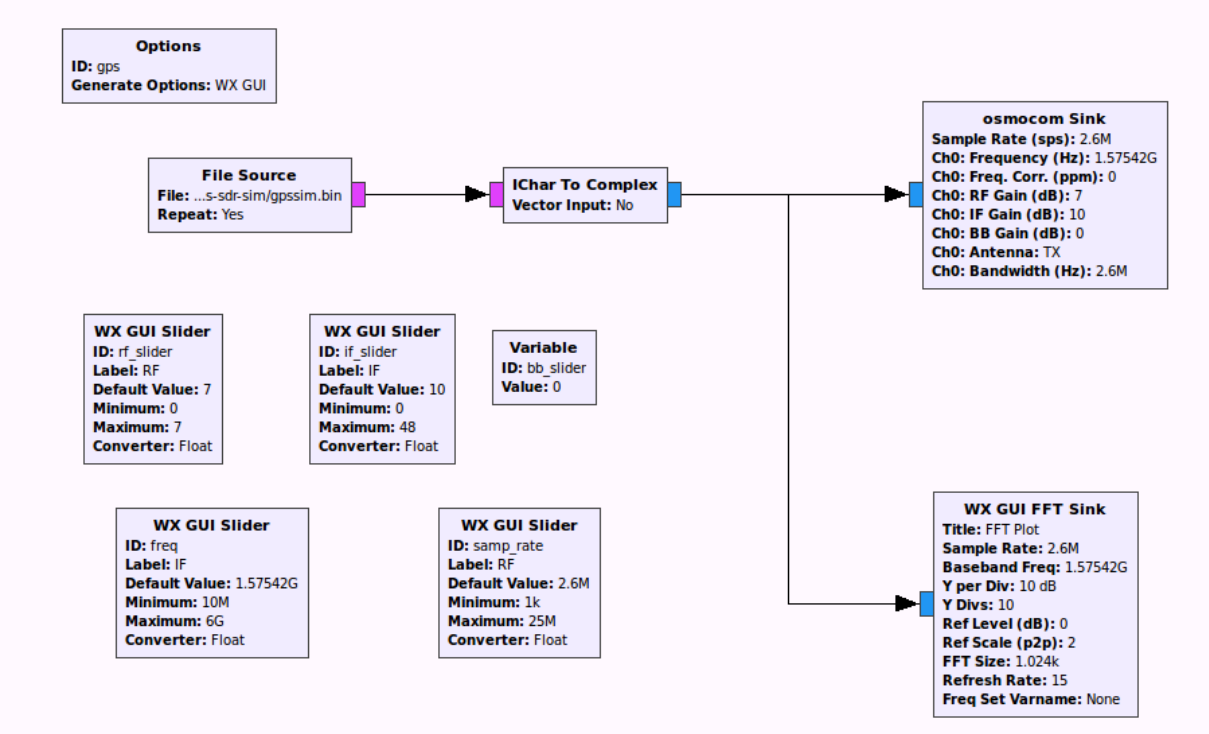

Figura 9

Diagrama de bloques GNU Radio. Transmisión GPS

Código Python generado en ANEXO B

Por último, para el desarrollo de las pruebas GPS Spoofing mediante el HackRF One se debe denotar que el dispositivo cuenta con un reloj interno de hasta 20ppm de error[16]. Este error afecta a la frecuencia de emisión en la señal de forma igual a la formula:

$$error_{freq} = \pm error_{reloj} * freq$$

Con una sencilla operación se puede calcular que trabajando en la banda L1 GPS, el error en la frecuencia de emisión seria:

$$error_{freq} = \pm \frac{20}{1000000} * 1575420000Hz = \pm 31508.4Hz$$

Esta desviación en la frecuencia de emisión impide la emisión en la frecuencia correcta, y por lo tanto impide que los dispositivos GPS reconozcan la señal simulada como una proveniente de satélites. Este desajuste se puede solventar reduciendo el error de reloj mediante de la sustitución del reloj de serie, por uno de menor tasa de error interno. En este caso se ha solucionado este fallo mediante el uso de un reloj por oscilación con temperatura controlada de cristal de cuarzo "NooElec Tiny TCXO" que tiene una tasa de error interno de 0.5ppm reduciendo así la desviación en frecuencia a unos 788Hz lo cual está dentro de lo admisible para pruebas GPS[16].

Figura 10

Imagen del reloj NooElec Tiny TCXO Fuente: Amazon

Durante las pruebas realizadas se consiguió de forma satisfactoria la decepción de la señal GPS de varios dispositivos móviles tanto de forma estática como dinámica (ANEXO D). Las limitaciones actuales del sistema fueron los 300s de duración de archivo de la decepción en dinámico, y en estático a partir de las 4/5h de duración de la decepción, debido a que el aumento de la temperatura interna del reloj aumentaba la tasa de error impidiendo así la continuación del engaño.

Por otro lado, durante la realización de pruebas en dispositivos militares se encontraron los siguientes problemas para que el objetivo fijase la posición[17]:

Exceso de potencia: Ya que la señal GPS que llega desde los satélites es de tan baja potencia, los receptores GPS más sofisticados están programados para ignorar aquellas señales provenientes de satélite que lleguen con demasiada fuerza[18].

Reloj interno: Algunos de los receptores GPS con los que se realizaron pruebas poseen un reloj interno de alta precisión con el que comparan la señal de reloj que envían los satélites con la propia, y si la

diferencia entre ellas supera los 10µs la descartan o entran en un modo recovery en el que compara ambas señales de reloj que supera las 4/5h que el dispositivo es capaz de mantener.

Posicionamiento inesperado: Similar al anterior, algunos dispositivos GPS están programados para detectar si se han producido cambios de posición de gran distancia (decenas de kilómetros) en intervalos de tiempo muy breves (segundos).

6. Integración de medios

Las limitaciones en cuanto a la potencia de emisión, y por lo tanto al alcance del dispositivo HackRF One, hacen que este no sea adecuado para las necesidades del ET, ya que en las distancias de trabajo tan pequeñas[6] del HackRF One las acciones ECM resultan inútiles. Por ello, en este último objetivo del proyecto, se persigue la integración de las capacidades de Jamming y GPS Spoofing obtenidas, dentro de los amplificadores de los que disponen las EP y de esta forma disminuir las carencias de potencia del dispositivo HackRF One.

6.1. Desarrollo

El proceso de emisión de señal a través de la EP implica a cuatro elementos distintos desde que se genera la señal hasta que esta es emitida. Primero la señal es emitida a través de un generador de señales, el cual crea una onda con ancho de banda y frecuencia definidas; después esta señal pasa mediante cable a un switch que redirige la señal a alguno de los amplificadores de la EP. En función de la señal que se quiere transmitir, la EP cuenta con varios amplificadores específicos, cada uno dedicados a una banda de frecuencias. Tras su paso por los amplificadores la señal se transmite hacia la antena correspondiente donde es irradiada.

Para la integración idónea de los medios SDR dentro de las EP del ejército, son estos dos últimos elementos (amplificadores y antenas) los que ofrecen las características básicas en las que los dispositivos HackRF One no son suficientes para el desarrollo de acciones ECM. Por ello, el propósito de la integración es la sustitución de la salida de señal de la EP, por la del dispositivo HackRF One en los amplificadores correspondientes a la emisión deseada.

El diseño del HackRF One permite la salida de la señal de transmisión a través de un conector estándar tipo SMA hembra con el que se puede acoplar a distintos tipos de antenas. Sin embargo, el puerto de entrada de señal de los distintos amplificadores de las EP tiene una entrada tipo n macho. Por lo tanto, es necesario el uso de un adaptador SMA macho a N hembra para la correcta incorporación de la señal. Además, como motivo de seguridad, la salida de señal del generador de la EP debe ser cerrada con una carga, en este caso de valor nominal de 50Ω, para evitar daños en el generador por estar el circuito abierto.

Otra posible limitación a tener en cuenta en el acoplado del HackRF en los amplificadores de la EP es la potencia de salida de la señal emitida. Según el manual del sistema, debido a limitaciones eléctricas en los amplificadores, la potencia de *input* máxima permitida por seguridad del sistema es de 1mW, con lo que es necesario el ajuste de la potencia de emisión en el dispositivo, a fin de evitar la sobrecarga de los componentes del amplificador.

[6] La niveles de potencia del HackRF One hacen inviable el GPS Spoofing a más de 200m y distancias aún menores para las acciones Jamming.

Una vez realizadas las modificaciones necesarias se puede proceder a la ejecución de acciones ECM mediante SDR a través de la EP. En la siguiente figura se muestra un experimento realizado a modo de prueba de concepto, en el cual se ejecutó una acción jamming a 16W de potencia en la que el receptor se encontraba a 1.2Km de distancia.

Figura 11

.Prueba de integración del dispositivo HackRF One en la estación EP del ejército.

En ella se puede observar, tanto en el espectrograma de la parte superior de la imagen como en la cascada de intensidad de señal en la parte inferior de la misma, la correcta emisión de la señal con un ancho de banda de 4MHz, que va desde la frecuencia 1.573GHz a la 1.577GHz, y un valor SNR de hasta 30Db en las frecuencias de mayor potencia.

6.2. Problemas de futuro

El desarrollo de las capacidades de los dispositivos SDR ha demostrado ser capaz de cumplir con, e incluso superar, las capacidades actuales del ET. Sin embargo, la novedad de este tipo de dispositivos hace que su potencial no esté completamente desarrollado. Además, el hecho de que la mayor parte de la innovación en esta industria sea a través de los aficionados que en su tiempo libre desarrollan programas para aprovechar las capacidades de los SDR, no hace mucho en favor de la evolución de esta tecnología. Esta inmadurez tecnológica provoca que se puedan observar, en el futuro próximo del desarrollo, tres problemas principales a resolver para la adecuada integración de los dos medios.

El primero y más próximo de los problemas de futuro observables es el uso de Hardware general y de bajo coste en la producción de la mayoría de los dispositivos SDR, el cual provoca limitaciones en las capacidades de los mismos. Ya se han observado, por ejemplo, las limitaciones del reloj interno del

dispositivo HackRF One a la hora de hacer medidas de decepción GPS o, inestabilidades de emisión en anchos de frecuencia altos en las acciones de perturbación Jamming. La causa de estos es que como se menciona previamente los dispositivos SDR están orientados a un mercado civil y de aficionados, con lo que no se buscan tanto dispositivos de estado del arte, sino dispositivos capaces de realizar un amplio rango de funciones a bajo nivel.

El siguiente problema previsto será el modo de integración actual del HackRF One en las EP. La realización actual de la integración se ve condicionada por la necesidad de un cambio manual en los cables que conectan la salida a los distintos amplificadores, en función de la banda de trabajo utilizada; esto supone una limitación a la hora del uso del programa de perturbación Jamming creado ya que limita la flexibilidad de cambio en la frecuencia de trabajo. Además, la necesidad de un operador realizando los cambios manualmente hace inútil la capacidad de los dispositivos SDR de ser operados de forma remota a través de enlace de red. Una posible solución a este problema sería el diseño de un switch de antenas similar al que tienen incorporado en su interior las EP, con el que seleccionar automáticamente el amplificador por el cual se quiere enviar la señal.

Por último, los dispositivos SDR no están diseñados con la idea de sufrir las cargas y condiciones a las que se someten los sistemas del ET, por ello, una de las ultimas dificultades a evaluar será la puesta a punto de las especificaciones de estos dispositivos para que cumplan con los estándares a los que se someten los materiales militares[7]. Esto se podría solucionar mediante el diseño y producción de dispositivo SDR que cumplan con los requisitos requeridos tanto de resistencia a impactos o condiciones ambientales, como especificaciones técnicas internas del Hardware. De esta forma, se podría crear un dispositivo especifico que solucionara al mismo tiempo las limitaciones del primer problema, aunque la creación de Hardware propio que se centra en un mercado de venta tan reducido provocaría una inflación en los precios del producto; eliminando así una de las ventajas de los dispositivos SDR convencionales frente a los sistemas militares que era el bajo coste de compra y mantenimiento.

[7] Los materiales militares tienen que pasar por una serie de pruebas de control de calidad más estrictas que aquellas del ámbito civil.

7. Conclusiones

Tras el desarrollo de este proyecto se han demostrado las capacidades que ofrecen los dispositivos SDR a las acciones de EW y más concretamente a las ECM. Equipos como el HackRF One ofrecen un amplio abanico posibilidades tanto en la perturbación, como en la decepción de receptores. Por desgracia, el mercado de aplicación de estos dispositivos y la persecución de un bajo coste de producción y venta ha provocado que las especificaciones y por lo tanto capacidades que ofrecen se vean limitadas en la calidad de las mismas.

Ejemplo de estas limitaciones se pueden observar en los resultados de los experimentos prácticos realizados, en los cuales, si bien la flexibilidad Software que los SDR ofrece, era capaz de configurar el dispositivo de forma que permite la acción Jamming en anchos de banda de hasta 28MHz, el procesador digital de señales no era capaz de seguir el ritmo de trabajo en la emisión y provocaba fluctuaciones en anchos de banda superiores a 16MHz. Otro ejemplo de este problema hallado ha sido la limitación en las pruebas del uso prolongado de acciones GPS Spoofing, en las que el reloj interno del dispositivo perdía precisión debido a cambios en el comportamiento del cristal de cuarzo que manda la señal de reloj con el aumento de su temperatura interna.

Aun teniendo en cuenta estas limitaciones, se ha demostrado que los dispositivos SDR tienen la capacidad de realizar las mismas funciones que los sistemas de ECM actuales del ET e incluso de añadir algunas adicionales como la decepción GPS. Asimismo, también se ha confirmado un cierto grado de compatibilidad y de cooperación entre ambos sistemas lo cual supone un aliciente para tener en cuenta en la toma de decisión del añadido de estos sistemas en las unidades de EW.

En definitiva, los dispositivos SDR han demostrado un alto potencial y flexibilidad de desarrollo en las acciones ECM, todo ello con unos costes despreciables en comparación con aquel de los sistemas del ET. Si bien es cierto que se encuentra en una etapa muy temprana de desarrollo, esta tecnología cuenta con una base civil de colaboradores aficionados que realizan investigaciones de acceso libre y de forma gratuita con la finalidad de generar nuevo software y analizar posibles nuevas aplicaciones de estos dispositivos.

En resumen, los SDR ofrecen grandes capacidades y potencial, son capaces de realizar acciones Jamming y GPS Spoofing con un gran nivel de flexibilidad, y son capaces de extender estas capacidades a los sistemas actuales del ET. Sin embargo, cabe destacar que esta tecnología se encuentra en un estado muy temprano de su desarrollo y que el grupo de usuarios al que está orientada no requiere de las especificaciones tanto de Hardware como de niveles de resistencia que los estándares del ejército exigen. Esto implica que es fácil asumir la necesidad una inversión de dinero para el desarrollo de dispositivos SDR específicos acordes a las necesidades de las unidades de EW del ejército.

8. Bibliografía

[1]. :“Ejército de tierra:.” [Online]. Available: http://www.ejercito.mde.es/unidades/Madrid/rew31/Organizacion/index.html. [Accessed: 14-Oct-2019].

[2] “Introduction to Software-Defined Radio - Technical Articles.” [Online]. Available: https://www.allaboutcircuits.com/technical-articles/introduction-to-software-defined-radio/. [Accessed: 28-Oct-2019].

[3] “HackRF One · mossmann/hackrf Wiki · GitHub.” [Online]. Available: https://github.com/mossmann/hackrf/wiki/HackRF-One. [Accessed: 25-Oct-2019].

[4] “Cómo funciona el sistema de posicionamiento GPS | AristaSur.” [Online]. Available: https://www.aristasur.com/contenido/como-funciona-el-sistema-de-posicionamiento-gps. [Accessed: 30-Oct-2019].

[5] “Estructura de las Señales del GPS.” [Online]. Available: http://haciaelespacio.aem.gob.mx/revistadigital/articul.php?interior=350. [Accessed: 10-Oct-2019].

[6] *How a GPS Receiver Gets a Lock*. Gpsinformation.net.

[7] *New Civil Signals*. .

[8] “HowToUse - GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/HowToUse. [Accessed: 01-Nov-2019].

[9] “About GNU Radio · GNU Radio.” [Online]. Available: https://www.gnuradio.org/about/. [Accessed: 25-Oct-2019].

[10] “Guided Tutorial Introduction - GNU Radio.” [Online]. Available: https://wiki.gnuradio.org/index.php/Guided_Tutorial_Introduction. [Accessed: 01-Nov-2019].

[11] “Como hacer Interferencias satelitales o Jamming de satélites.” [Online]. Available: https://noticiasseguridad.com/tecnologia/como-hacer-interferencias-satelitales-o-jamming-de-satelites/. [Accessed: 25-Oct-2019].

[12] R. Kharadi and H. Mehta, “Audio File Transmission using GNU RADIO and USRP EEC 687-Mobile Computing (Fall 2016) Final Project Report.”

[13] T. Senator, P. Leahy, and D. Investigation, “HackRF One,” no. Lcdi, 2017.

[14] “GitHub - FrankBuss/gps-sdr-sim: Software-Defined GPS Signal Simulator.” [Online]. Available: https://github.com/FrankBuss/gps-sdr-sim. [Accessed: 10-Oct-2019].

[15] “Free GPS NMEA Simulator software.” [Online]. Available: https://www.labsat.co.uk/index.php/en/free-gps-nmea-simulator-software. [Accessed: 28-Oct-2019].

[16] “GPS Simulator - SDR - Software Defined Radio - Hak5 Forums.” [Online]. Available: https://forums.hak5.org/topic/38290-gps-simulator/. [Accessed: 30-Oct-2019].

[17] “How to deal with GPS jamming and spoofing - CRFS - Spectrum Monitoring and Geolocation.” [Online]. Available: https://www.crfs.com/blog/how-to-deal-with-gps-jamming-and-spoofing/. [Accessed: 03-Nov-2019].

[18] “Anti-Spoofing | GPS Lab.” [Online]. Available: https://gps.stanford.edu/research/current-research/anti-spoofing. [Accessed: 03-Nov-2019].

ANEXO A

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Fm Trx
# Generated: Sat Oct 12 02:55:16 2019
##################################################

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from gnuradio import analog
from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import filter
from gnuradio import gr
from gnuradio.eng_option import eng_option
from gnuradio.filter import firdes
from gnuradio.wxgui import forms
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import osmosdr
import time
import wx


class fm_trx(grc_wxgui.top_block_gui):

    def __init__(self):
        grc_wxgui.top_block_gui.__init__(self, title="Fm Trx")
        _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
        self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))

        ##################################################
        # Variables
        ##################################################
        self.audio_samp = audio_samp = 44100
        self.samp_rate = samp_rate = 2600000
        self.quad_rate = quad_rate = audio_samp*2
```

```
self.if_gain = if_gain = 25
self.freq = freq = 225000000

##################################################
# Blocks
##################################################
_samp_rate_sizer = wx.BoxSizer(wx.VERTICAL)
self._samp_rate_text_box = forms.text_box(
    parent=self.GetWin(),
    sizer=_samp_rate_sizer,
    value=self.samp_rate,
    callback=self.set_samp_rate,
    label='samp_rate',
    converter=forms.float_converter(),
    proportion=0,
)
self._samp_rate_slider = forms.slider(
    parent=self.GetWin(),
    sizer=_samp_rate_sizer,
    value=self.samp_rate,
    callback=self.set_samp_rate,
    minimum=1000000,
    maximum=28000000,
    num_steps=100,
    style=wx.SL_HORIZONTAL,
    cast=float,
    proportion=1,
)
self.Add(_samp_rate_sizer)
_if_gain_sizer = wx.BoxSizer(wx.VERTICAL)
self._if_gain_text_box = forms.text_box(
    parent=self.GetWin(),
    sizer=_if_gain_sizer,
    value=self.if_gain,
    callback=self.set_if_gain,
    label='if_gain',
    converter=forms.float_converter(),
    proportion=0,
)
self._if_gain_slider = forms.slider(
    parent=self.GetWin(),
    sizer=_if_gain_sizer,
    value=self.if_gain,
    callback=self.set_if_gain,
    minimum=1,
    maximum=47,
    num_steps=100,
    style=wx.SL_HORIZONTAL,
    cast=float,
    proportion=1,
)
self.Add(_if_gain_sizer)
```

```
        _freq_sizer = wx.BoxSizer(wx.VERTICAL)
        self._freq_text_box = forms.text_box(
            parent=self.GetWin(),
            sizer=_freq_sizer,
            value=self.freq,
            callback=self.set_freq,
            label='freq',
            converter=forms.float_converter(),
            proportion=0,
        )
        self._freq_slider = forms.slider(
            parent=self.GetWin(),
            sizer=_freq_sizer,
            value=self.freq,
            callback=self.set_freq,
            minimum=10000000,
            maximum=5000000000,
            num_steps=100,
            style=wx.SL_HORIZONTAL,
            cast=float,
            proportion=1,
        )
        self.Add(_freq_sizer)
        self.rational_resampler_xxx_0 = filter.rational_resampler_ccc(
              interpolation=samp_rate,
              decimation=quad_rate,
              taps=None,
              fractional_bw=None,
        )
        self.osmosdr_sink_0 = osmosdr.sink( args="numchan=" + str(1) + " " + "" )
        self.osmosdr_sink_0.set_sample_rate(samp_rate)
        self.osmosdr_sink_0.set_center_freq(freq, 0)
        self.osmosdr_sink_0.set_freq_corr(0, 0)
        self.osmosdr_sink_0.set_gain(44, 0)
        self.osmosdr_sink_0.set_if_gain(if_gain, 0)
        self.osmosdr_sink_0.set_bb_gain(0, 0)
        self.osmosdr_sink_0.set_antenna("", 0)
        self.osmosdr_sink_0.set_bandwidth(0, 0)

        self.blocks_multiply_const_vxx_0 = blocks.multiply_const_vff((0.00003, ))
        self.blocks_file_source_0 = blocks.file_source(gr.sizeof_float*1, "/opt/sigintos/Fm-
Transmitter/griffin-21.wav", True)
        self.analog_wfm_tx_0 = analog.wfm_tx(
            audio_rate=audio_samp,
            quad_rate=quad_rate,
            tau=75e-6,
            max_dev=75e3,
        )

        ##################################################
        # Connections
        ##################################################
```

```
        self.connect((self.analog_wfm_tx_0, 0), (self.rational_resampler_xxx_0, 0))
        self.connect((self.blocks_file_source_0, 0), (self.blocks_multiply_const_vxx_0, 0))
        self.connect((self.blocks_multiply_const_vxx_0, 0), (self.analog_wfm_tx_0, 0))
        self.connect((self.rational_resampler_xxx_0, 0), (self.osmosdr_sink_0, 0))

    def get_audio_samp(self):
        return self.audio_samp

    def set_audio_samp(self, audio_samp):
        self.audio_samp = audio_samp
        self.set_quad_rate(self.audio_samp*2)

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self._samp_rate_slider.set_value(self.samp_rate)
        self._samp_rate_text_box.set_value(self.samp_rate)
        self.osmosdr_sink_0.set_sample_rate(self.samp_rate)

    def get_quad_rate(self):
        return self.quad_rate

    def set_quad_rate(self, quad_rate):
        self.quad_rate = quad_rate

    def get_if_gain(self):
        return self.if_gain

    def set_if_gain(self, if_gain):
        self.if_gain = if_gain
        self._if_gain_slider.set_value(self.if_gain)
        self._if_gain_text_box.set_value(self.if_gain)
        self.osmosdr_sink_0.set_if_gain(self.if_gain, 0)

    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self._freq_slider.set_value(self.freq)
        self._freq_text_box.set_value(self.freq)
        self.osmosdr_sink_0.set_center_freq(self.freq, 0)


def main(top_block_cls=fm_trx, options=None):
    if gr.enable_realtime_scheduling() != gr.RT_OK:
        print "Error: failed to enable real-time scheduling."

    tb = top_block_cls()
    tb.Start(True)
```

```
    tb.Wait()


if __name__ == '__main__':
    main()
```

ANEXO B

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
##################################################
# GNU Radio Python Flow Graph
# Title: Gps
# Generated: Sat Oct 12 02:53:27 2019
##################################################

if __name__ == '__main__':
    import ctypes
    import sys
    if sys.platform.startswith('linux'):
        try:
            x11 = ctypes.cdll.LoadLibrary('libX11.so')
            x11.XInitThreads()
        except:
            print "Warning: failed to XInitThreads()"

from gnuradio import blocks
from gnuradio import eng_notation
from gnuradio import gr
from gnuradio import wxgui
from gnuradio.eng_option import eng_option
from gnuradio.fft import window
from gnuradio.filter import firdes
from gnuradio.wxgui import fftsink2
from gnuradio.wxgui import forms
from grc_gnuradio import wxgui as grc_wxgui
from optparse import OptionParser
import osmosdr
import time
import wx


class gps(grc_wxgui.top_block_gui):

    def __init__(self):
        grc_wxgui.top_block_gui.__init__(self, title="Gps")
        _icon_path = "/usr/share/icons/hicolor/32x32/apps/gnuradio-grc.png"
        self.SetIcon(wx.Icon(_icon_path, wx.BITMAP_TYPE_ANY))
```

```
##################################################
# Variables
##################################################
self.samp_rate = samp_rate = 2600000
self.rf_slider = rf_slider = 7
self.if_slider = if_slider = 10
self.freq = freq = 1575420000
self.bb_slider = bb_slider = 0

##################################################
# Blocks
##################################################
_samp_rate_sizer = wx.BoxSizer(wx.VERTICAL)
self._samp_rate_text_box = forms.text_box(
    parent=self.GetWin(),
    sizer=_samp_rate_sizer,
    value=self.samp_rate,
    callback=self.set_samp_rate,
    label="RF",
    converter=forms.float_converter(),
    proportion=0,
)
self._samp_rate_slider = forms.slider(
    parent=self.GetWin(),
    sizer=_samp_rate_sizer,
    value=self.samp_rate,
    callback=self.set_samp_rate,
    minimum=1000,
    maximum=25000000,
    num_steps=100,
    style=wx.SL_HORIZONTAL,
    cast=float,
    proportion=1,
)
self.Add(_samp_rate_sizer)
_rf_slider_sizer = wx.BoxSizer(wx.VERTICAL)
self._rf_slider_text_box = forms.text_box(
    parent=self.GetWin(),
    sizer=_rf_slider_sizer,
    value=self.rf_slider,
    callback=self.set_rf_slider,
    label="RF",
    converter=forms.float_converter(),
    proportion=0,
)
self._rf_slider_slider = forms.slider(
    parent=self.GetWin(),
    sizer=_rf_slider_sizer,
    value=self.rf_slider,
    callback=self.set_rf_slider,
    minimum=0,
    maximum=7,
```

```
        num_steps=1,
        style=wx.SL_HORIZONTAL,
        cast=float,
        proportion=1,
    )
    self.Add(_rf_slider_sizer)
    _if_slider_sizer = wx.BoxSizer(wx.VERTICAL)
    self._if_slider_text_box = forms.text_box(
        parent=self.GetWin(),
        sizer=_if_slider_sizer,
        value=self.if_slider,
        callback=self.set_if_slider,
        label="IF",
        converter=forms.float_converter(),
        proportion=0,
    )
    self._if_slider_slider = forms.slider(
        parent=self.GetWin(),
        sizer=_if_slider_sizer,
        value=self.if_slider,
        callback=self.set_if_slider,
        minimum=0,
        maximum=48,
        num_steps=48,
        style=wx.SL_HORIZONTAL,
        cast=float,
        proportion=1,
    )
    self.Add(_if_slider_sizer)
    _freq_sizer = wx.BoxSizer(wx.VERTICAL)
    self._freq_text_box = forms.text_box(
        parent=self.GetWin(),
        sizer=_freq_sizer,
        value=self.freq,
        callback=self.set_freq,
        label="IF",
        converter=forms.float_converter(),
        proportion=0,
    )
    self._freq_slider = forms.slider(
        parent=self.GetWin(),
        sizer=_freq_sizer,
        value=self.freq,
        callback=self.set_freq,
        minimum=10000000,
        maximum=6000000000,
        num_steps=1000,
        style=wx.SL_HORIZONTAL,
        cast=float,
        proportion=1,
    )
    self.Add(_freq_sizer)
```

```
        self.wxgui_fftsink2_0 = fftsink2.fft_sink_c(
            self.GetWin(),
            baseband_freq=1575420000,
            y_per_div=10,
            y_divs=10,
            ref_level=0,
            ref_scale=2.0,
            sample_rate=samp_rate,
            fft_size=1024,
            fft_rate=15,
            average=False,
            avg_alpha=None,
            title="FFT Plot",
            peak_hold=False,
        )
        self.Add(self.wxgui_fftsink2_0.win)
        self.osmosdr_sink_0 = osmosdr.sink( args="numchan=" + str(1) + " " + "" )
        self.osmosdr_sink_0.set_sample_rate(samp_rate)
        self.osmosdr_sink_0.set_center_freq(freq, 0)
        self.osmosdr_sink_0.set_freq_corr(0, 0)
        self.osmosdr_sink_0.set_gain(rf_slider, 0)
        self.osmosdr_sink_0.set_if_gain(if_slider, 0)
        self.osmosdr_sink_0.set_bb_gain(bb_slider, 0)
        self.osmosdr_sink_0.set_antenna("TX", 0)
        self.osmosdr_sink_0.set_bandwidth(samp_rate, 0)

        self.blocks_interleaved_char_to_complex_0 = blocks.interleaved_char_to_complex(False)
        self.blocks_file_source_0 = blocks.file_source(gr.sizeof_char*1,
"/media/sigintos/b09d94fd-127b-4697-ba28-8cb22b1b121d/home/sigintos/Downloads/gps-
sdr-sim/gpssim.bin", True)

        ##################################################
        # Connections
        ##################################################
        self.connect((self.blocks_file_source_0, 0), (self.blocks_interleaved_char_to_complex_0,
0))
        self.connect((self.blocks_interleaved_char_to_complex_0, 0), (self.osmosdr_sink_0, 0))
        self.connect((self.blocks_interleaved_char_to_complex_0, 0), (self.wxgui_fftsink2_0, 0))

    def get_samp_rate(self):
        return self.samp_rate

    def set_samp_rate(self, samp_rate):
        self.samp_rate = samp_rate
        self.osmosdr_sink_0.set_sample_rate(self.samp_rate)
        self.osmosdr_sink_0.set_bandwidth(self.samp_rate, 0)
        self.wxgui_fftsink2_0.set_sample_rate(self.samp_rate)
        self._samp_rate_slider.set_value(self.samp_rate)
        self._samp_rate_text_box.set_value(self.samp_rate)

    def get_rf_slider(self):
        return self.rf_slider
```

```
    def set_rf_slider(self, rf_slider):
        self.rf_slider = rf_slider
        self._rf_slider_slider.set_value(self.rf_slider)
        self._rf_slider_text_box.set_value(self.rf_slider)
        self.osmosdr_sink_0.set_gain(self.rf_slider, 0)

    def get_if_slider(self):
        return self.if_slider

    def set_if_slider(self, if_slider):
        self.if_slider = if_slider
        self._if_slider_slider.set_value(self.if_slider)
        self._if_slider_text_box.set_value(self.if_slider)
        self.osmosdr_sink_0.set_if_gain(self.if_slider, 0)

    def get_freq(self):
        return self.freq

    def set_freq(self, freq):
        self.freq = freq
        self._freq_slider.set_value(self.freq)
        self._freq_text_box.set_value(self.freq)
        self.osmosdr_sink_0.set_center_freq(self.freq, 0)

    def get_bb_slider(self):
        return self.bb_slider

    def set_bb_slider(self, bb_slider):
        self.bb_slider = bb_slider
        self.osmosdr_sink_0.set_bb_gain(self.bb_slider, 0)


def main(top_block_cls=gps, options=None):

    tb = top_block_cls()
    tb.Start(True)
    tb.Wait()


if __name__ == '__main__':
    main()
```

K

ANEXO C

0.0, -3813477.954, 3554276.552, 3662785.237

0.1, -3813477.599, 3554276.226, 3662785.918

0.2, -3813477.240, 3554275.906, 3662786.598

0.3, -3813476.876, 3554275.590, 3662787.278

0.4, -3813476.508, 3554275.280, 3662787.958

0.5, -3813476.135, 3554274.975, 3662788.638

0.6, -3813475.757, 3554274.675, 3662789.318

0.7, -3813475.375, 3554274.381, 3662789.997

0.8, -3813474.988, 3554274.091, 3662790.676

0.9, -3813474.597, 3554273.807, 3662791.355

1.0, -3813474.201, 3554273.528, 3662792.033

1.1, -3813473.800, 3554273.255, 3662792.711

1.2, -3813473.396, 3554272.986, 3662793.388

1.3, -3813472.986, 3554272.723, 3662794.065

1.4, -3813472.573, 3554272.466, 3662794.741

1.5, -3813472.155, 3554272.213, 3662795.416

1.6, -3813471.733, 3554271.967, 3662796.091

1.7, -3813471.306, 3554271.725, 3662796.764

1.8, -3813470.875, 3554271.489, 3662797.438

1.9, -3813470.440, 3554271.258, 3662798.110

2.0, -3813470.001, 3554271.033, 3662798.781

2.1, -3813469.557, 3554270.814, 3662799.452

2.2, -3813469.110, 3554270.599, 3662800.121

2.3, -3813468.658, 3554270.391, 3662800.790

2.4, -3813468.202, 3554270.187, 3662801.457

2.5, -3813467.742, 3554269.990, 3662802.123

2.6, -3813467.278, 3554269.798, 3662802.788

2.7, -3813466.810, 3554269.611, 3662803.452

2.8, -3813466.338, 3554269.430, 3662804.114

2.9, -3813465.862, 3554269.254, 3662804.776

3.0, -3813465.383, 3554269.084, 3662805.436

3.1, -3813464.899, 3554268.920, 3662806.094

3.2, -3813464.412, 3554268.761, 3662806.751

3.3, -3813463.920, 3554268.608, 3662807.407

3.4, -3813463.425, 3554268.461, 3662808.061

3.5, -3813462.927, 3554268.319, 3662808.713

3.6, -3813462.424, 3554268.183, 3662809.364

3.7, -3813461.918, 3554268.053, 3662810.013

3.8, -3813461.409, 3554267.928, 3662810.660

3.9, -3813460.895, 3554267.809, 3662811.306

4.0, -3813460.379, 3554267.695, 3662811.950

4.1, -3813459.858, 3554267.587, 3662812.592

4.2, -3813459.335, 3554267.485, 3662813.232

4.3, -3813458.807, 3554267.389, 3662813.870

4.4, -3813458.277, 3554267.298, 3662814.506

4.5, -3813457.743, 3554267.213, 3662815.140

4.6, -3813457.205, 3554267.134, 3662815.772

4.7, -3813456.665, 3554267.061, 3662816.402

4.8, -3813456.121, 3554266.993, 3662817.030

4.9, -3813455.574, 3554266.931, 3662817.655

5.0, -3813455.023, 3554266.875, 3662818.278

Anexo D

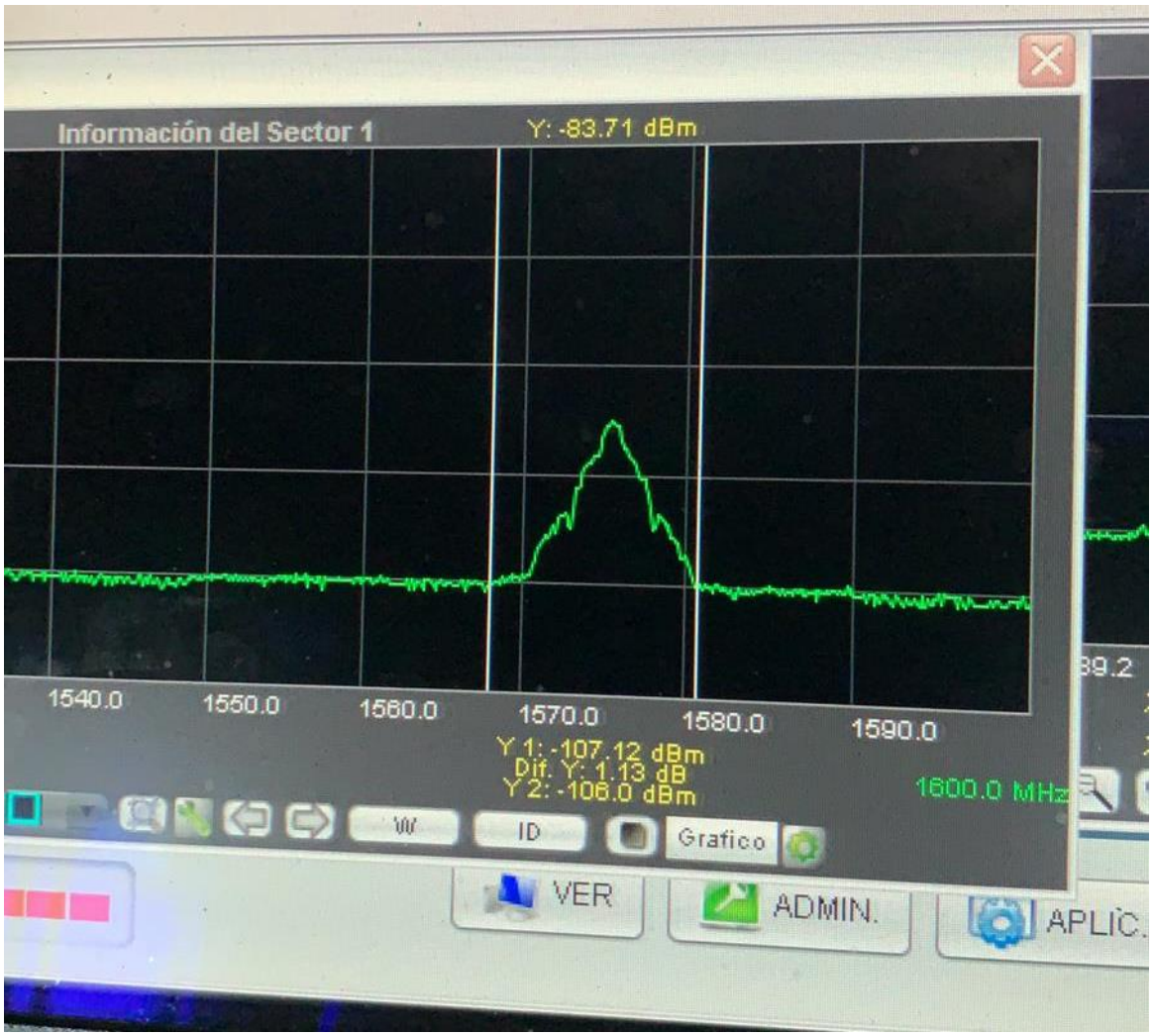

Figura 12

Muestra de perturbación en la banda GPS con un SNR de 25 Db y un ancho de banda 12MHz a 16 W de potencia

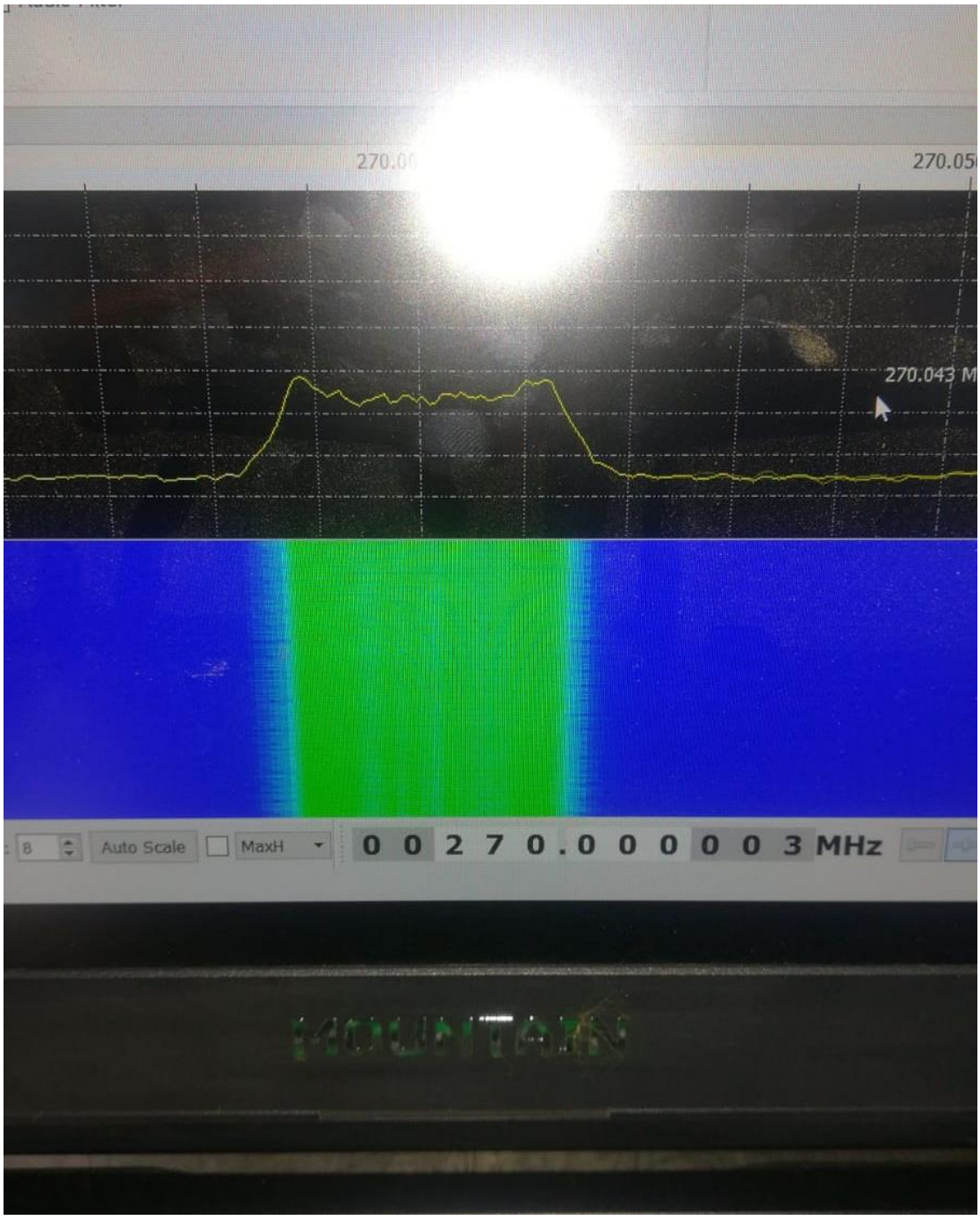

Figura 13

Muestra de perturbación de 10 km de distancia con un ancho de banda de 1 MHz y un SNR de 27 Db con 16 W de potencia

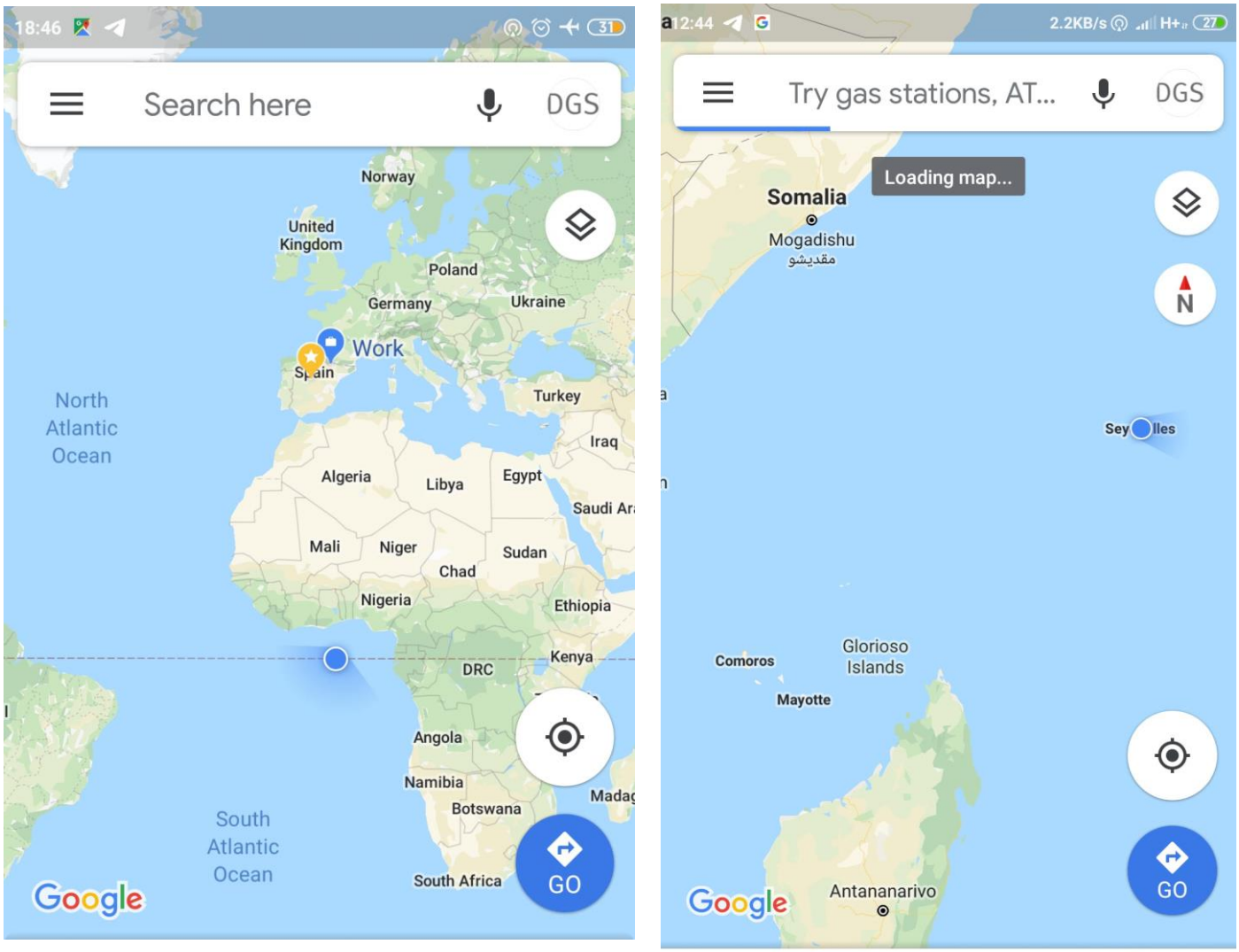

Figura 14

Muestra de resultados obtenidos tanto en GPS Spoofing estático (izquierda) con en GPS Spoofing Dinámico (derecha)